NED University of Engineering and Technology

Mechanical Engineering

Project Report

Restaurant Delivery Time Prediction using Python

Analysis & Prediction

Course : Machine Learning (CT-562)

Project Advisor : Dr. Haider Ali

Presenter : Rafay Hasan (ME-009)
Presenter : Danish Akhund

Contents

Food Delivery Time Prediction using Python

Introduction:

More businesses are moving online these days, and consumers are ordering online instead of traveling to the store to buy. Food Panda popular online platforms for ordering food products. They provide food delivery options. If the order is complete, a partner will pick up and deliver the meal to the given address via a delivery service. In online food-ordering businesses, delivery time is critical. As a result, estimated food delivery time prediction to reach the buyer's location is critical. The LSTM neural network is one of the methods that may be implemented in this circumstance. Come on, let's study the LSTM models in detail.

Objectives of Food Delivery Time Prediction

1. Make an accurate estimate of when the food will arrive, thus increasing customer confidence.

2. Plan delivery routes and driver schedules more efficiently by predicting how many orders will arrive so that delivery providers can use their resources better.

3. Make deliveries faster by looking at past delivery data and determining the attributes that affect them.

4. Grow business because of buyer satisfaction with the speed of delivery.

Based on these goals, we will use the LSTM Neural Network to develop a model that can estimate the delivery time of orders accurately based on the age of the delivery partner, the partner's rating, and the distance between the restaurant and the buyer's place.

Step 1: Import Library

Pandas and NumPy libraries are used together for data analysis. NumPy provides fast mathematical functions for multidimensional arrays, while Pandas makes it easier to analyze and manipulate data with more complex data structures like DataFrame and Series. Meanwhile, the Plotly Express library makes it easy for users to create interactive visualizations in Python. It can use minimal code to

create various charts, such as scatter plots, line charts, bar charts, and maps. The Sequential class is a type of model in Keras that allows users to create a neural network by adding layers to it in sequential order. Then, Dense and LSTM are to create layers in the Keras model and also customize their configurations.

```
import pandas as pd
import numpy as np
import plotly.express as px

data = pd.read_csv("deliverytime.txt")
print(data.head())
```

```
     ID Delivery_person_ID  Delivery_person_Age  Delivery_person_Ratings  \
0  4607     INDORES13DEL02                   37                      4.9
1  B379     BANGRES18DEL02                   34                      4.5
2  5D6D     BANGRES19DEL01                   23                      4.4
3  7A6A    COIMBRES13DEL02                   38                      4.7
4  70A2     CHENRES12DEL01                   32                      4.6

   Restaurant_latitude  Restaurant_longitude  Delivery_location_latitude  \
0            22.745049             75.892471                   22.765049
1            12.913041             77.683237                   13.043041
2            12.914264             77.678400                   12.924264
3            11.003669             76.976494                   11.053669
4            12.972793             80.249982                   13.012793

   Delivery_location_longitude Type_of_order Type_of_vehicle  Time_taken(min)
0                    75.912471        Snack      motorcycle                24
1                    77.813237        Snack         scooter                33
2                    77.688400       Drinks      motorcycle                26
3                    77.026494       Buffet      motorcycle                21
4                    80.289982        Snack         scooter                30
```

Step 2: Read the Data

The availability of data is crucial to any data analysis task. It is essential to have a dataset that contains all the required features and variables for the particular task at hand. The dataset given here is a cleaned version of the original dataset on Kaggle.

Dataset Acquisition

Food Delivery Time Prediction: Case Study

JANUARY 2, 2023

Download the dataset below to solve this Data Science case study on food delivery time prediction. (Dataset Source: **Kaggle**)

Download Data

Food Delivery Time Prediction: Case Study

Predicting the delivery time of your order is a challenging task for every food delivery service like Zomato and Swiggy.

One of the best strategies to predict the delivery time is by calculating the distance between the point of picking up the order and the point of delivering the order. And then predicting the delivery time based on how much time your delivery partners took to deliver orders in the past for the same distance.

1. ID: order ID number
2. Delivery_person_ID: ID number of the delivery partner
3. Delivery_person_Age: Age of the delivery partner
4. Delivery_person_Ratings: ratings of the delivery partner based on past deliveries
5. Restaurant_latitude: The latitude of the restaurant
6. Restaurant_longitude: The longitude of the restaurant
7. Delivery_location_latitude: The latitude of the delivery location
8. Delivery_location_longitude: The longitude of the delivery location
9. Type_of_order: The type of meal ordered by the customer
10. Type_of_vehicle: The type of vehicle delivery partner rides
11. Time_taken(min): The time taken by the delivery partner to complete the order

You are required to predict the delivery time based on the distance covered by the delivery partner to deliver the order.

Let's see detailed information about the dataset we use with the info() command.

```
In [2]: # columns information
        data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45593 entries, 0 to 45592
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   ID                           45593 non-null  object 
 1   Delivery_person_ID           45593 non-null  object 
 2   Delivery_person_Age          45593 non-null  int64  
 3   Delivery_person_Ratings      45593 non-null  float64
 4   Restaurant_latitude          45593 non-null  float64
 5   Restaurant_longitude         45593 non-null  float64
 6   Delivery_location_latitude   45593 non-null  float64
 7   Delivery_location_longitude  45593 non-null  float64
 8   Type_of_order                45593 non-null  object 
 9   Type_of_vehicle              45593 non-null  object 
 10  Time_taken(min)              45593 non-null  int64  
dtypes: float64(5), int64(2), object(4)
memory usage: 3.8+ MB
```

Now let’s have a look at whether this dataset contains any null values or not:

data.isnull().sum()

```
In [3]: # finding Null values
        data.isnull().sum()

Out[3]: ID                             0
        Delivery_person_ID             0
        Delivery_person_Age            0
        Delivery_person_Ratings        0
        Restaurant_latitude            0
        Restaurant_longitude           0
        Delivery_location_latitude     0
        Delivery_location_longitude    0
        Type_of_order                  0
        Type_of_vehicle                0
        Time_taken(min)                0
        dtype: int64
```

The dataset does not have any null values. Let’s move further!

Step 3: Haversine Formula

The Haversine formula is used to find the distance between two geographical locations. The formula refers to this Wikipedia page as follows:

The Haversine formula:

a = sin²(Δlat/2) + cos(lat1) * cos(lat2) * sin²(Δlon/2)
c = 2 * atan2(√a, √(1−a))
d = R * c

where:

- lat1 and lat2 are the latitudes of the two points
- Δlat means the difference between the latitudes
- Δlon means the difference between the longitudes
- R means the radius of the sphere
- d means the distance between the two points in the same units as R

It takes the latitude and longitude of two points and converts the angles to radians to perform the necessary calculations. We use this formula because the dataset doesn’t provide the distance between the restaurant and the delivery location. There are only latitude and longitude. So, let’s calculate it and then create a distance column in the dataset.

Below is how we can find the distance between the restaurant and the delivery location based on their latitudes and longitudes by using the haversine formula:

```python
R = 6371

# Convert degrees to radians
def deg_to_rad(degrees):
    return degrees * (np.pi/180)

# Function to calculate the distance between two points using the haversine formula
def distcalculate(lat1, lon1, lat2, lon2):
    d_lat = deg_to_rad(lat2-lat1)
    d_lon = deg_to_rad(lon2-lon1)
    a = np.sin(d_lat/2)**2 + np.cos(deg_to_rad(lat1)) * np.cos(deg_to_rad(lat2)) * np.sin(d_lon/2)**2
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1-a))
    return R * c
```

The parameter “lat” means latitude, and “lon” means longitude. The deg_to_rad function is helpful for converting degrees to radians. At the same time, calculate the distance between two location points using the variables a1 and a2. The variable stores the result of multiplying a1 and a2, while the c variable stores the result of the Haversine formula calculation, which produces the distance between the two location points.

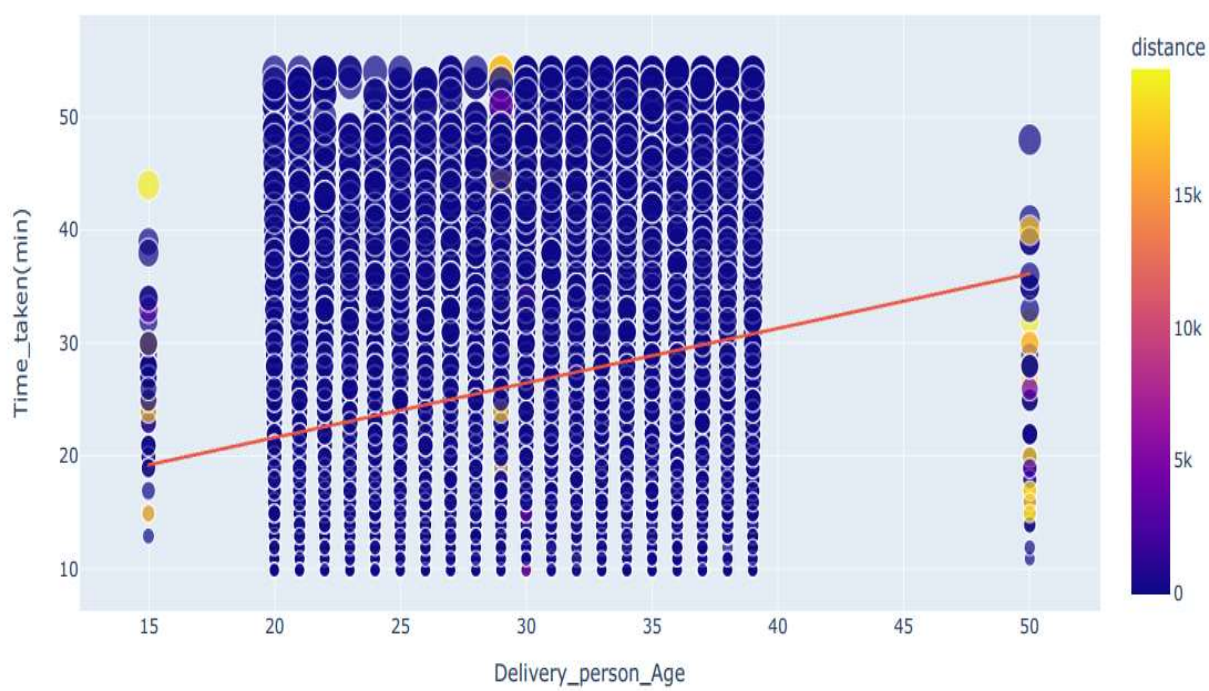
Data Exploration

We have added a distance column to the dataset. Now, we will analyze the effect of distance and delivery time.

So, explore the data to find relationships between the features.

First Relationships b/w Time & Age of rider

```
figure = px.scatter(data_frame = data,
                    x="Delivery_person_Age",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    color = "distance",
                    trendline="ols",
                    title = "Relationship Between Time Taken and Age")
figure.show()
```



There is a consistent relationship between the time taken and the distance travelled to deliver the food. It means that most delivery partners deliver food within 25-30 minutes, regardless of distance. The graph shows faster food delivery when partners are younger than their older counterparts.

<u>Second explore the correlation between delivery time and delivery partner ratings</u>.

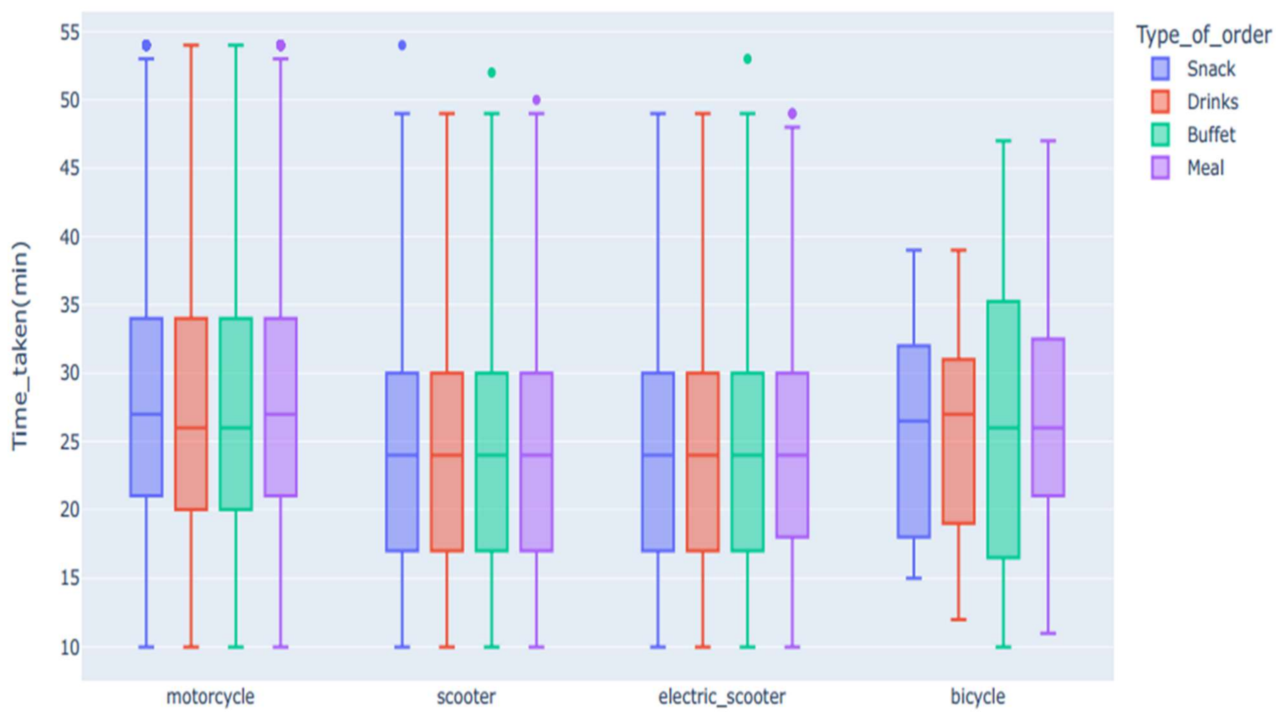
```
figure = px.scatter(data_frame = data,
                    x="Delivery_person_Ratings",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    color = "distance",
                    trendline="ols",
                    title = "Relationship Between Time Taken and Ratings")
figure.show()
```

There is an inverse linear relationship between the time taken to deliver the food and the ratings of the delivery partner. It means delivery partners with higher ratings take less time to deliver the food compared to partners with low ratings.

The next step will be to see whether the delivery partner's vehicle affects the delivery time or not. Means

<u>Relation b/w Type of Order , Vehicle & Time</u>

```
fig = px.box(data,
             x="Type_of_vehicle",
             y="Time_taken(min)",
             color="Type_of_order")
fig.show()
```



The graph shows that the type of delivery partner's vehicle and the type of food delivered do not significantly affect delivery time.

Through the analysis above, we can determine that the delivery partner's age, the delivery partner's rating, and the distance between the restaurant and the

delivery location are the features that have the most significant impact on food delivery time.

Step 4: Build an LSTM Model and Make Predictions

Previously, we have determined three features that significantly affect the time taken, namely the delivery partner's age, the delivery partner's rating, and distance. So the three features will become independent variables (x), while the time taken will become the dependent variable (y).

we need to train an LSTM neural network to predict food delivery time. The aim is to create a precise model that uses features like distance, delivery partner age, and rating to estimate food delivery time. The trained model can then be used to predict new data points or unseen scenarios.

```
from sklearn.model_selection import train_test_split
x = np.array(data[["Delivery_person_Age","Delivery_person_Ratings","distance"]])
y = np.array(data[["Time_taken(min)"]])
xtrain, xtest, ytrain, ytest = train_test_split(x, y,test_size=0.10,random_state=42)
```

```
from keras.models import Sequential
from keras.layers import Dense, LSTM
model = Sequential()
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (xtrain.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 3, 128)            66560

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 25)                1625

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 117619 (459.45 KB)
Trainable params: 117619 (459.45 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

The code block above explains: The first line starts building the model architecture by creating an instance of the Sequential class. The following three lines define the layers of the model. The first layer is an LSTM layer with 128 units, which returns sequences and takes input for shape (xtrain.shape[1], 1). Here, xtrain is the input training data, and shape[1] represents the number of features in the input data. The return_sequences parameter is set to True because there will be more layers after this one. The second layer is also an LSTM layer, but with 64 units and return_sequences set to False, indicating that this is the last layer. The third line adds a dense layer with 25 units, which reduces the output of the LSTM layers to a more manageable size. Finally, the fourth line adds a dense layer with one unit, which is the output layer of the model.

Now let's train the previously created model.

```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(xtrain, ytrain, batch_size=1, epochs=9)
```

```
Epoch 1/9
41033/41033 [==============================] - 365s 8ms/step - loss: 69.4178
Epoch 2/9
41033/41033 [==============================] - 357s 9ms/step - loss: 63.4860
Epoch 3/9
41033/41033 [==============================] - 355s 9ms/step - loss: 61.3758
Epoch 4/9
41033/41033 [==============================] - 355s 9ms/step - loss: 60.6749
Epoch 5/9
41033/41033 [==============================] - 363s 9ms/step - loss: 59.6832
Epoch 6/9
41033/41033 [==============================] - 360s 9ms/step - loss: 59.7502
Epoch 7/9
41033/41033 [==============================] - 384s 9ms/step - loss: 59.8624
Epoch 8/9
41033/41033 [==============================] - 387s 9ms/step - loss: 59.2573
Epoch 9/9
36427/41033 [=========================>....] - ETA: 41s - loss: 59.4699
```

The 'adam' parameter is a popular optimization algorithm for deep learning models, and the
'mean_squared_error' parameter is a common loss function used in regression problems. The parameter batch_size = 1 means that the model will update its weights after each sample is processed during training. The epochs parameter is set to 9, meaning the model will be trained on the entire dataset for nine iterations.

Finally, let's test the model's performance for predicting food delivery times given three input parameters (delivery partner age, delivery rating, and distance).

```
print("Food Delivery Time Prediction")
a = int(input("Age of Delivery Partner: "))
b = float(input("Ratings of Previous Deliveries: "))
c = int(input("Total Distance: "))

features = np.array([[a, b, c]])
print("Predicted Delivery Time in Minutes = ", model.predict(features))
```

```
Age of Delivery Partner: 26
Ratings of Previous Deliveries: 5
Total Distance: 25
1/1 [==============================] - 91s 91s/step
Predicted Delivery Time in Minutes =  [[25.485943]]
```

The given result is a prediction of the delivery time for a hypothetical food delivery order based on the
trained LSTM neural network model using the following input features:

Delivery Partner's Age: 26

Previous Delivery Ratings: 5.0

Total distance: 25

The output of the prediction is shown as "Delivery Time Prediction in Minutes = [[25.48]]," which
means that the model has estimated that the food delivery will take approximately 25.48 minutes to reach the destination.

Conclusion

This article starts by calculating the distance between the restaurant and the delivery location. Then, it
analyzes previous delivery times for the same distance before predicting food delivery times in real-time
using LSTM. Broadly speaking, in this post, we have discussed the following:

1. How to calculate the distance using the haversine formula?
2. How to find the features that affect the food delivery time prediction?
3. How to use LSTM neural network model to predict the food delivery time?

References:

- Food Delivery Time Prediction with LSTM Neural Network Download Share crown iconAta Amrullah — Published On April 8, 2023 and Last Modified On May 12th, 2023
- “290+ Machine Learning Projects with Python | by Aman Kharwal | Coders Camp | Medium.”
- “Food_Delivery_Time_Analysis_Project/MiniProject_FoodDelivery_Final.Ipynb at 08a0ac7a21c47e5f6f967b0453d0037e41e3e9c5 · Dev26git/Food_Delivery_Time_Analysis_Project.”
- Kharwal, “Food Delivery Time Prediction Using Python | Aman Kharwal.”
- Débora Guimarães de Sousa Using Machine Learning to Predict
  On-Time Delivery Metropolia University of Applied Sciences
  Master’s Degree Degree Programme in Business Informatics Master’s
  Thesis,30/11/2022
- Supervised Learning for Arrival Time Estimations in Restaurant Meal Delivery, Preprint in Transportation Science · August 2020, Florentin Hildebrandt , Otto-von-Guericke-Universität Magdeburg
- Food Delivery Time Prediction: How Does It Work?
  It’s not just Google Maps…but it is a little bit Google Maps.
  Written by Jye Sawtell-Rickson
  Published on May. 31, 2022